VB.NET - REGULAR EXPRESSIONS

 Copyright © tutorialspoint.com

A **regular expression** is a pattern that could be matched against an input text. The .Net framework provides a regular expression engine that allows such matching. A pattern consists of one or more character literals, operators, or constructs.

Constructs for Defining Regular Expressions

There are various categories of characters, operators, and constructs that lets you to define regular expressions. Click the follwoing links to find these constructs.

- Character escapes
- Character classes
- Anchors
- Grouping constructs
- Quantifiers
- Backreference constructs
- Alternation constructs
- Substitutions
- Miscellaneous constructs

The Regex Class

The Regex class is used for representing a regular expression.

The Regex class has the following commonly used methods:

S.N	Methods & Description
1	**Public Function IsMatch (input As String) As Boolean** Indicates whether the regular expression specified in the Regex constructor finds a match in a specified input string.
2	**Public Function IsMatch (input As String, startat As Integer) As Boolean** Indicates whether the regular expression specified in the Regex constructor finds a match in the specified input string, beginning at the specified starting position in the string.
3	**Public Shared Function IsMatch (input As String, pattern As String) As Boolean** Indicates whether the specified regular expression finds a match in the specified input string.
4	**Public Function Matches (input As String) As MatchCollection** Searches the specified input string for all occurrences of a regular expression.
5	**Public Function Replace (input As String, replacement As String) As String** In a specified input string, replaces all strings that match a regular expression pattern with a specified

	replacement string.
6	**Public Function Split (input As String) As String()** Splits an input string into an array of substrings at the positions defined by a regular expression pattern specified in the Regex constructor.

For the complete list of methods and properties, please consult Microsoft documentation.

Example 1

The following example matches words that start with 'S':

```
Imports System.Text.RegularExpressions
Module regexProg
   Sub showMatch(ByVal text As String, ByVal expr As String)
      Console.WriteLine("The Expression: " + expr)
      Dim mc As MatchCollection = Regex.Matches(text, expr)
      Dim m As Match
      For Each m In mc
          Console.WriteLine(m)
      Next m
   End Sub
   Sub Main()
      Dim str As String = "A Thousand Splendid Suns"
      Console.WriteLine("Matching words that start with 'S': ")
      showMatch(str, "\bS\S*")
      Console.ReadKey()
   End Sub
End Module
```

When the above code is compiled and executed, it produces following result:

```
Matching words that start with 'S':
The Expression: \bS\S*
Splendid
Suns
```

Example 2

The following example matches words that start with 'm' and ends with 'e':

```
Imports System.Text.RegularExpressions
Module regexProg
   Sub showMatch(ByVal text As String, ByVal expr As String)
      Console.WriteLine("The Expression: " + expr)
      Dim mc As MatchCollection = Regex.Matches(text, expr)
      Dim m As Match
      For Each m In mc
          Console.WriteLine(m)
      Next m
   End Sub
   Sub Main()
      Dim str As String = "make a maze and manage to measure it"
      Console.WriteLine("Matching words that start with 'm' and ends with 'e': ")
      showMatch(str, "\bm\S*e\b")
      Console.ReadKey()
   End Sub
End Module
```

When the above code is compiled and executed, it produces following result:

```
Matching words start with 'm' and ends with 'e':
The Expression: \bm\S*e\b
```

```
make
maze
manage
measure
```

Example 3

This example replaces extra white space:

```
Imports System.Text.RegularExpressions
Module regexProg
   Sub Main()
      Dim input As String = "Hello    World   "
      Dim pattern As String = "\\s+"
      Dim replacement As String = " "
      Dim rgx As Regex = New Regex(pattern)
      Dim result As String = rgx.Replace(input, replacement)
      Console.WriteLine("Original String: {0}", input)
      Console.WriteLine("Replacement String: {0}", result)
      Console.ReadKey()
   End Sub
End Module
```

When the above code is compiled and executed, it produces following result:

```
Original String: Hello   World
Replacement String: Hello World
```